70th International Astronautical Congress (IAC), Washington D.C., United States, 21-25 October 2019.

IAC-19.D1.1.2

The Development of Dynamic Guidance and Navigation Algorithms for Autonomous On-Orbit Multi-Satellite Aggregation

David A. Barnhart[1], Ryan H. Duong[2], Lizvette Villafaña[3], Jaimin Patel[4], Shreyash Annapureddy[5]

[1]*Department of Astronautical Engineering, University of Southern California Information Sciences Institute and Space Engineering Research Center, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292,* barnhart@isi.edu
[2]*Department of Astronautical Engineering, University of Southern California Information Sciences Institute and Space Engineering Research Center, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292,* rduong@usc.edu
[3]*Department of Aerospace Engineering, University of Southern California Information Sciences Institute and Space Engineering Research Center, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292,* lvillafa@isi.edu
[4]*Department of Computer Science, University of Southern California Information Sciences Institute and Space Engineering Research Center, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292,* jaiminpa@usc.edu
[5]*Department of Computer Science, University of Southern California Information Sciences Institute and Space Engineering Research Center, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292,* sannapur@usc.edu

Abstract

With the growing popularity of low-cost satellite missions merged with the ability to potentially rendezvous and "connect" on orbit, the potential to enable mission objectives typically reserved for very large monolithic satellites with many small low-cost satellites arises not just in swarm flight but through the possibility of on-orbit satellite "aggregation". Ultimately, such satellite aggregation concepts, resulting in a new singular spacecraft comprised from many individual functioning elements, must adhere to basic spacecraft design considerations like attitude control and thermal management upon aggregation/integration. When considering on-orbit operations, an autonomous methodology must be implemented to ensure seamless spacecraft functionality and control through aggregation. The University of Southern California Information Sciences Institute (USC ISI) and Space Engineering Research Center (SERC) have fabricated multiple modular pseudo-satellite prototypes and developed an autonomous GNC reconfiguration algorithm to redefine a singular reaction control system from multiple contributing spacecraft. These prototypes, bounded by 3 degrees of freedom on a frictionless air-bearing table, perform individual maneuvers with 8 body mounted thrusters to simulate rendezvous and proximity operations (RPO) docking and aggregation. Upon aggregation an integrated algorithm demonstrates autonomous reconfiguration of the aggregate platforms thrusters and execute additional maneuvers with its new control orientation. In parallel, the SERC team created a simulation environment, which enables additional virtual aggregation to show scale of the algorithm's capability in optimizing aggregated GNC subsystem for up to "N" number of elements. This paper will present preliminary results of the first set of algorithmic experimentation in active autonomous GNC reconfiguration.

Keywords: Aggregation, GNC, cellular morphology

Acronyms/Abbreviations

3 Degrees of Freedom 3DOF
Center of gravity ..... CG
Computer Aided Design ..... CAD
Extended Kalman Filter ..... EKF
Guidance, Navigation, and Control ..... GNC
Linear Time Invariant ..... LTI
Model Reference Adaptive Control ..... MRAC
Moments of Inertia ..... MOI
Pulse Width Modulation ..... PWM
Reliable User Datagram Protocol ..... RUDP
Rendezvous & Proximity Operations ..... RPO
Transmission Control Protocol ..... TCP
User Datagram Protocol ..... UDP

1 Introduction

Within the last 5-10 years, significant change has occurred within the space industry, transitioning from Government domination to private commercial organizations who see space differently. From the amazing new discoveries of inhabitable planets by the Kepler telescope to Virgin Galactic's successful orbital flights and its waiting list of people worldwide to fly, there is a new energy and excitement not seen for 25 years. This era of "New Space" is opening up access for millions of people who never considered space accessible or useful to them in any form. However, these new space companies and countries have not changed the space systems they are fielding; today's satellites look much like the ones built from 50 years ago. Major systems and subsystems are combined in the same way whether in the Hubble Space Telescope, a large geostationary communications satellite, or today's Cubesats. The size of elements, components, and subsystems may change to accommodate the final satellite, but the fundamental makeup of spacecraft resources of power, propulsion, attitude control, etc., is no different. Today, mass has become a proxy in the search for lower cost solutions to accommodate shrinking budgets, but that comes with a requisite consequent exchange of performance. The historical equation of cost as a direct function of mass drives this solution. However, what if this cost-mass-performance equation can be broken? What if these limitations in performance associated with size could be ameliorated or even avoided by the aggregation of elements? What if aggregation could be done on orbit?

A new domain in space systems – cellular space platform morphology, based on biological inspiration has been created to explore this new approach [2–4]. The concept utilizes "cellular" morphology to change satellites from large monolithic entities into platforms built from hundreds of small low cost "satlets". In essence, the ability to literally "transform" an electro-mechanical system on orbit, to allow unlimited aggregation and assembly of large platforms, may enable new platforms with high power, propulsion, and exploration capabilities in space in ways not possible before.

This on-orbit "cellularization" concept is meant to create the means for a satellite or space platform to literally aggregate, reconfigure and deaggregate itself on command, and allows for on-orbit "assembly" to grow any size or volume platform, much like biological entities use discrete cells multiplied in the thousands or millions to create complex highly functional bodies. However, one major challenge to do this is creation of unique software that allows the "cells" to share their resources and capabilities seamlessly.

2 Aggregation Architecture

To begin characterization and description of what a new computational architecture for aggregation requires, we detail a notional top-level concept in the depiction shown in Figure 1. Any depiction must consider the physical and virtual layers which exist with every single element in an aggregation, as well as the various transport layers for data and control required to enable ubiquitous autonomous aggregation capability.

For the purpose of this paper and first step in postulating a new computational architecture, Figure 1 depicts a layering of functionality. Each is defined below.

1. **Software Layer:** At the lowest level, or the software inside a component layer (with a component with its own processor or controller), we consider the characterization of internal (i.e., cell, component, element) set of capabilities that inherently exist and the "needs" that are required outside the cell/component/element for it to function (i.e., power). This can be considered as inside the cell or component that can support other cells or systems, or can be shared with other cells/systems. In typical space systems an "event scheduler" exists that runs the physical hardware systems (i.e., cell or component or element) based on a single clock or hardware processing cycle from the main processor (e.g., 100 Hz). Tasks then represent the mechanism which sends out capabilities to the rest of the cells, or bring in needs that the cells or system is asking for. Again, this is usually performed by a single central processing system.

2. **Hardware Layer:** At the next layer, or "hardware layer", we envision this to represent where the physical aggregation of various elements occurs. The elements can be what we have defined as "satlets", each having its own inherent "capabilities" (i.e., a processor, an IMU, a set amount of power available etc.), as well as something that does not have as much virtual processing capability like a solar array, a propulsion element, a payload, etc. This is the layer where the physical connections are made between the elements, where the cell/component software layer is embedded in each of the processing elements (which defines their capabilities and needs).

3. **Data Transport Layer:** To enable aggregation across multiple physical systems, the next layer becomes a data transport layer. This could be as simple as a wireless protocol that is used to "connect" the processing based elements to each other and transport data (i.e.,

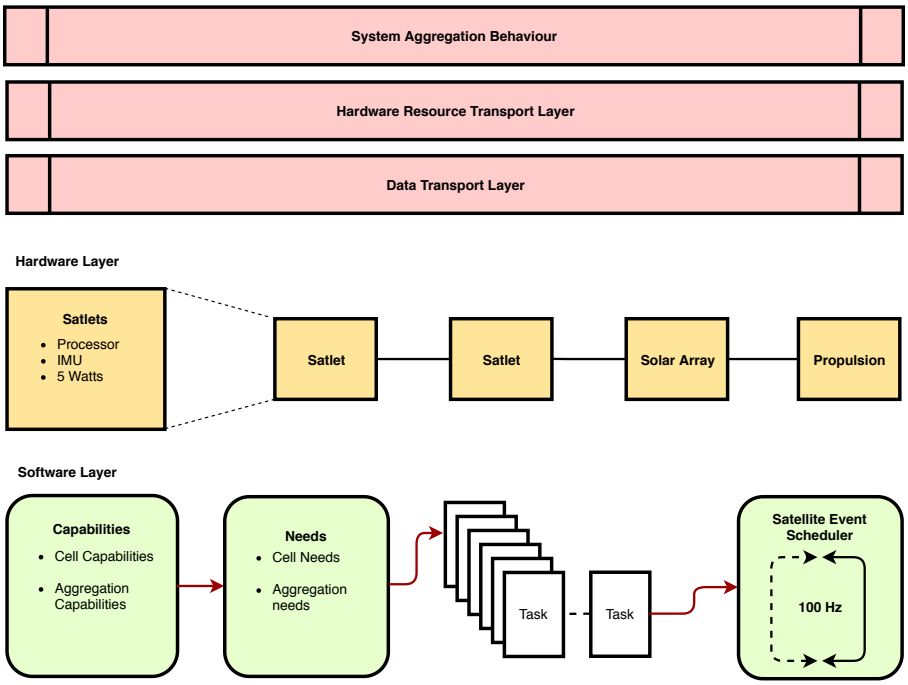


Fig. 1: Notional Aggregation Computational Architecture

information that is packetized) from each element to all the others.

4. **Hardware Resource Transport Layer:** The next layer could be represented as a separate entity or as a part of the data transport layer. We define this layer as the hardware resource transport layer, where the needs and capabilities of each hardware element are shared with others and each cell or element are permitted to operate together.

5. **System Aggregation Behaviour:** The last layer to enable the aggregated systems to function as a monolithic entity, would be the "system aggregation behaviour" level, which has code that takes advantage of the hardware and data transport layers to create behaviours of the aggregated system. As an example of this level, the aggregate system's thermal response would be evaluated, then individual commands would be sent to elements with passive or active thermal systems to support the entire assembly as a whole to be isothermal. Another example considers the system's GNC. The active aggregated GNC behavioural function would respond to the CG and MOI changes of the aggregated system and would send commands to each cell/component/element to exercise the appropriate GNC hardware.

2.1 Aggregation Considerations - Hardware

Enabling aggregation requires two elements, the hardware itself that will physically "connect" and the supporting software that controls and operates the specific hardware. Previously, we described an "architecture" that includes data and resource transport layers that could provide overarching control and communications to the cells (i.e., components, elements, satlets, etc.). Now, let us examine key considerations to implement actual aggregation, with respect to both hardware and operational software.

Aggregation can occur on either a heterogeneous or homogeneous spectrum. Figure 2 shows a graphical identification of that spectrum, where potentially aggregatable cells can either be "system level" or pure "component level". Today all satellite hardware is component level, tied together with a central processing system and custom code to operate each component. In an aggregated architecture, each component becomes a cell with capabilities and needs as previously discussed. An aggregated system should be able to merge any level of hetero or homogeneity component/cell/element ubiquitously, without concern for new code being written or what its operation requires.

Hardware examples of aggregatable elements that represent

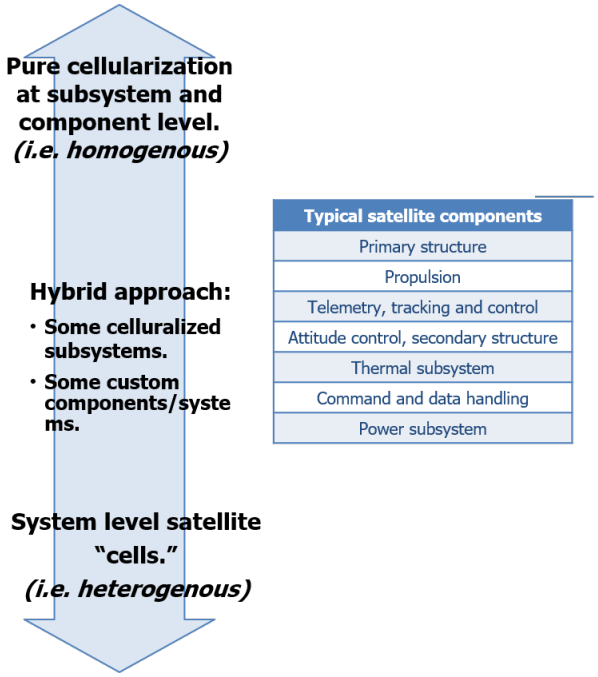


Fig. 2: Cellularization can occur through a spectrum of heterogeneity and homogeneity.

the spectrum are shown in Figure 3.

Hardware aggregation then requires physical connectivity between all the elements. Specialized interfaces for both legacy and new components would be required, for which the two hardware instantiations shown in the Figure above have been developed (cited here [7] [8]).

2.2 *Aggregation Considerations - Software*

As described in the previous section, software is currently bespoke to the hardware that is being used, and thus highly rigid. To extend the capabilities to allow aggregation, the software must be robust enough to manage the hardware resources and their corresponding constraints at both an individual/component and an aggregation level.

The desired architectural goal is to have an autonomous distributed computational architecture across individuals that once aggregated behave as a single entity. Thus, it is necessary for the aggregation computational architecture to have an automated resource management system. The aggregation must facilitate typical spacecraft functions with respect to the coordination and allocation of physical and computational resources. This resource management strategy must also consider the complexities of having a distributed system of interacting components.

To achieve scalablility with the aggregation's distributed resources, there must be either protocols or management systems enabling resources sharing. For example, the computational resource manager may delegate tasks to free resources, allowing for efficient scheduling and execution of tasks across different individuals/components within the aggregation.

In this distributed system it is important to clarify two types of computational architectures, centralized and decentralized. In a centralized architecture, coordination of the aggregation is managed by a single master entity with all others behaving as slaves. All information is processed by the central authority, which determines the aggregation's behaviour. As the aggregation grows in size, the centralized architecture must be able to handle the increased workload, including receiving and processing data from the entire aggregation. An advantage of a centralized architecture lies in its simplicity, yet a severe disadvantage is the lack of redundancy if the master fails. In a decentralized architecture, the increased redundancy ensures robustness of the aggregation in the event of any failures, but also greatly increases the complexity of software architecture. Information sharing is not handled by any single component, rather all components are aware of the aggregation state or will request information which is relevant from sources distributed in the aggregation.

2.2.1 *Component Level Behaviour Control*

This is the internal operation of the hardware. Some may have control (i.e., processors), others may be controlled by a separate processor that is attached to it. As an example, an electric propulsion system may have a processor on board to control the power processing unit and flow rates of the propellant.

2.2.2 *System Level Behaviour (The Aggregation Level)*

System level behaviour may be unique software that merges the needs/capabilities of hundreds of individual elements together, at one time. An example here would be an overarching algorithm that can coordinate a number of electric propulsion subsystems that integrates their firing time to conform to lateral thrust, or a maneuver.

3 First example of aggregation behaviour software-GNC

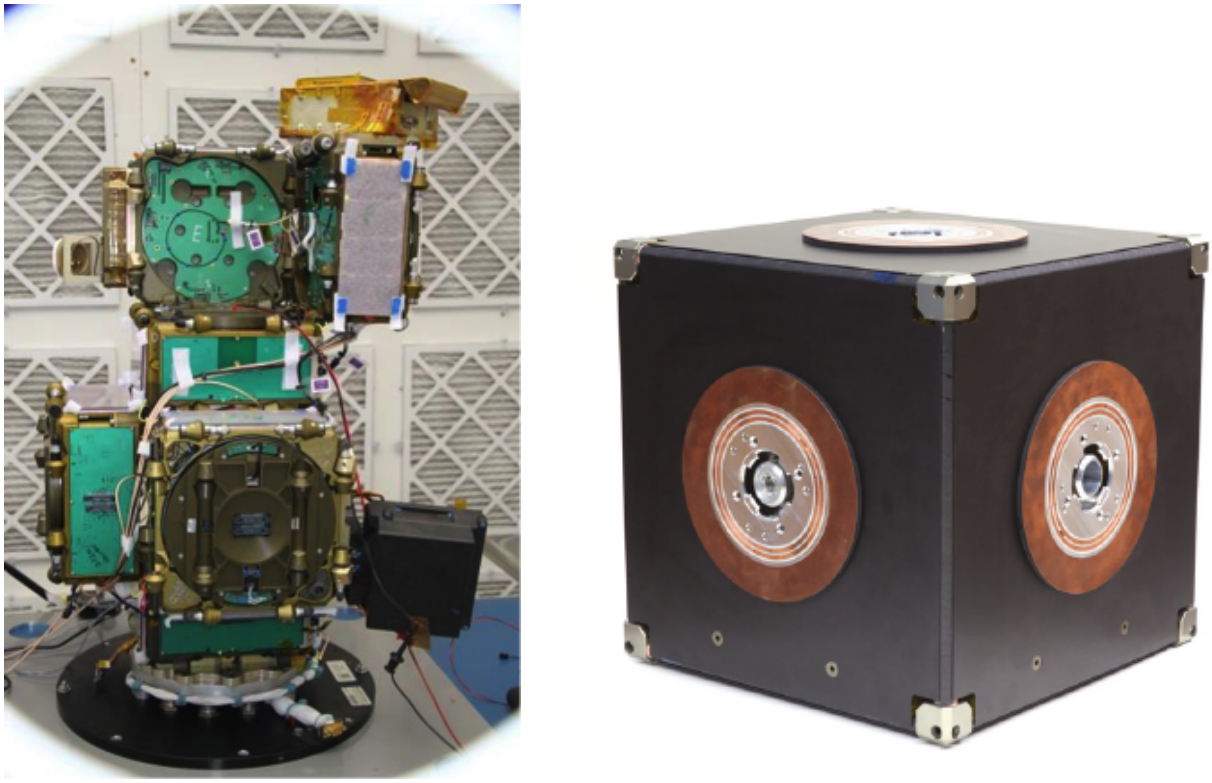
Researchers at the SERC developed pseudo-satellite prototypes, dubbed "Satbots," with a supporting computational architecture to demonstrate a first application of multi-satellite aggregation. The initial instantiation focused on the concept of merging multiple independent GNC subsystems to operate as an aggregated platform.

3.1 Satbot Prototype Design

The Satbot is built to emulate orbit operations on a ground testbed (Figure 4). An on-board compressed air tank feeds through three flat-round air bearings at 60 psi to lift the Satbot platform 5 microns on top of a float glass platen, establishing a near-frictionless environment resulting in 3DOF. Compressed air is also redirected through eight unidirectional output solenoid valves at 100 psi to support the GNC subsystem. The current Satbot design places these solenoid valves at the edge of its circular platform, maximizing the ability to generate torque about the center of mass at the Satbot's geometric center. For future reference in this paper, the terms "actuator" or "thruster" represent these solenoid valves. The Satbot holds two docking ports for aggregation and RPO demonstration. Additionally, each Satbot carries a Wi-Fi enabled Odroid-C2 single-board computer with a Linux Operating System to support GNC software and sensor operations. An example of proposed Satbot aggregation is shown in Figure 5.

3.2 Satbot Proposed GNC Operations

The proposed concept of operations of the Satbots is meant to demonstrate the potential of autonomous GNC and multi-spacecraft aggregation. Each has continuous communication between the other Satbots, thus each will determine the nec-



(a) (b)

Fig. 3: **(a) NovaWurks implementation of a "system satlet", called "HISAT's" . (b) Module from iBoss Inc., which is an implementation of a heterogeneous module**

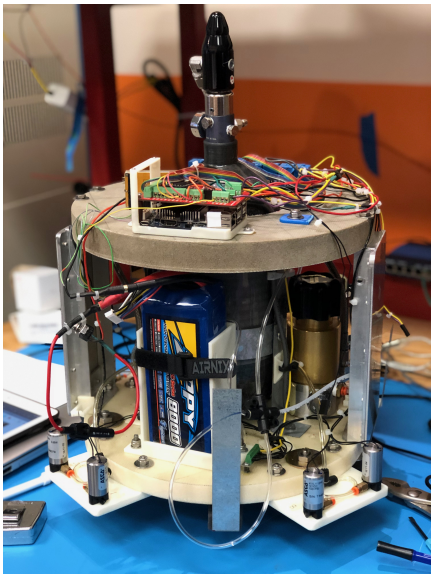


Fig. 4: The current Satbot design.

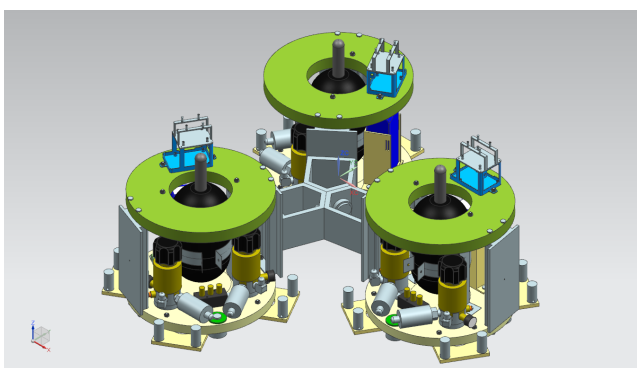


Fig. 5: An example of an aggregate system consisting of three Satbots.

essary states which drives the GNC subsystems.

Still in its early development phase, the Satbot is manually provided the desired states to perform GNC. Onboard sensors measure the navigational state, which is stored for guidance and control calculations. Sensor data is passed through a Kalman filter for state estimation. After calculating the error, or difference between the desired and estimated state, the controller determines the necessary forces and torques required and then the corresponding firing time for each thruster. This GNC operation continues until the Satbot achieves the desired state.

To support aggregation, the relative knowledge between all Satbots must be known at some level. Therefore, each Satbot possess an uploaded "identification file," which is a set of hardware parameters relative to the Satbot's body frame. If the Satbot identifies that a docking face is active, such that an aggregate system exists, then each Satbot transfers its parameters into the data transport layer of the computational architecture to create an aggregation identification file to support the system's GNC. To simulate a docking event (i.e., aggregation event), we create a new reconfiguration algorithm, which imports the appropriate Satbot parameters from the architecture's data transport layer to calculate the new aggregate properties. The algorithm feeds this information back into *one* of the Satbots, and GNC operation continues for the aggregate system. As the aggregate system is intended to operate with the proposed decentralized computational architecture, each Satbot possesses the capability to carry on aggregated GNC for the system to operate as a single monolithic entity.

For the Satbot prototype an autonomous distributed resource management system is not required. Rather, in this Satbot prototype demonstration, a single "Master" Satbot runs the GNC algorithm written in C. Positional data received from all the aggregated satbots is received and processed using an EKF and the GNC algorithm performs the necessary calculations to create the aggregation firing time matrix. The other aggregated Satbots run a separate "Slave" code which receives the firing times and fires thrusters for the given interval. This implementation provides a simple mechanism to test the GNC code for up to N-number of Satbots, provided the communication network allows all firing times to fire thrusters concurrently. Each of the Satbots has the capability to behave as the "Master" or "Slave". Each of the guidance, navigation and control elements relative to our demonstration is described in greater detail in the following sections.

3.3 Guidance and Navigation

In the SERC labs, each Satbot uses a Pozyx tag, a real-time location system electronics board that measures the navigation state. Determining the positional state relies on multilateration, the methodology of GPS radio-navigation. Like GPS, four unique "anchors," or Pozyx modules that serve as fixed reference points, transmit an ultra-wide band signal to the Pozyx tag to locate its position with an accuracy down to 10 cm on the testbed. Onboard the Pozyx tag lies a 3-axis accelerometer and a 3-axis gyroscope, providing the capability to measure accelerations and orientation. Each Satbot measures the vector presented in Equation 1 and creates the state vector as seen in Equation **??**. The state is then

processed through an EKF to determine the state estimate and its associated covariance matrix.

$$\vec{y} = [x, y, q_1, q_2, q_3, q_4, a_x, a_y, \omega_z]^{\mathrm{T}} \tag{1}$$

The orientation represented via quaternions, where $q_{1,2,3}$ represents the quaternion vector components and q_4 represents the quaternion weight. Accelerations are denoted as a_x and a_y respectively, where the subscript corresponds to a particular direction. Lastly, the angular velocity is represented by ω_z. In regards to the state vector, $\dot{x}$ and $\dot{y}$ represent velocity in x and y, respectively and θ represents rotation about the z-axis.

Provided the desired state, the Satbot determines the deviations from its trajectory, which is sent to the controller.

3.4 Control Scheme

A controller is implemented to determine the required control inputs necessary for the system to reach the desired state based on current deviations from the ideal trajectory.

Though a variety of controllers exist, only a few are deemed suitable to support the concept of aggregation and autonomous GNC, where stable performance is maintained for potentially varying system dynamics from physical aggregation. Upon aggregation, the controller designed for the individual cells will need to adapt in order for the aggregated system to remain autonomous. Without adaptation, the control system designed for one individual cell will prove ineffective, as it was designed for a system with different mass properties. Adaptive controls have been studied for the pasts decades and are useful for systems with any sort of uncertainty. In the case of an aggregated satellite system, the controller must be compatible for any system of N aggregated cells.

Simulations provide a rapid way to design different adaptive control algorithms and are usually used for stability analysis due to the lack of existing stability metrics, compared to traditional control theory. A co-simulation using Simulink for the control design and NX Motion for system dynamics was used to build and validate the controller, and will be discussed later.

3.4.1 Simple PD Controller

Since we currently focus on demonstrating the development of the reconfiguration algorithm (discussed later), we designed a simple Proportional-Derivative (P-D) controller for the first initial stages of testing. With the state deviations and predetermined controller gains, Equations 2 to 4 calculate the control input vector. [9]

$$F_x = K_{p,position}\delta_x + K_{d,position}\delta_{\dot{x}} \tag{2}$$
$$F_y = K_{p,position}\delta_y + K_{d,position}\delta_{\dot{y}} \tag{3}$$
$$T_z = K_{p,attitude}\delta_\theta + K_{d,attitude}\delta_{\dot{\theta}} \tag{4}$$

The K terms represent the controller gains, where the subscripts p and d correspond to the proportional and derivative terms respectively. The δ terms correspond to the state deviations as designated by its subscript.

However, in addressing the aggregation behaviour, the selected controller will not maintain stability and performance for large scale configurations. Each Satbot addition changes the system response of the controller and will eventually drive the controller to instability. This fact influences the controller design to consider the implementation of adaptive control.

3.4.2 Model Reference Adaptive Control (MRAC)

As a first look at an "adaptive" control methodology we evaluated the MRAC design.

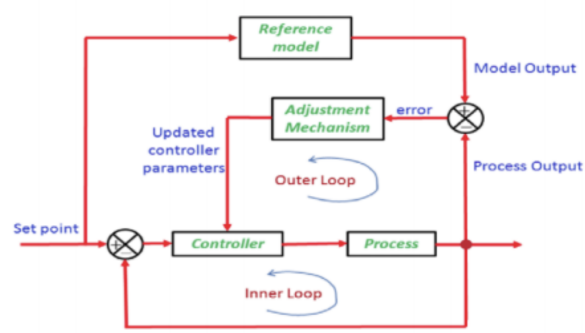


Fig. 6: Block diagram of a generalized MRAC scheme. [1]

As seen in Figure 6, a typical MRAC consists of an additional outer loop, compared to a traditional controller with (one)

feedback loop. In this outer loop, the state estimates are compared to reference model outputs, rather than a desired state. This error is then fed into an adjustment mechanism in which the on-board processor estimates the control gains that are then combined with a predetermined control law to calculate the system's control inputs. In this way, the system's controller is continuously updating and adapting to system dynamics.

The reference model selected reflects desired system behaviour and typically is formulated as a LTI model. [10] For an aggregated satellite system consisting of N Satbots with a nonlinear plant modeled as:

$$\dot{\vec{x}}^{\text{agg. cm}} = \mathbf{A} f(\vec{x}) + \mathbf{B}\vec{u} \tag{5}$$

and a reference model: [10]

$$\dot{\vec{x}}_m^{agg.cm} = \mathbf{A_m}\vec{x}_m + \mathbf{B_m}\vec{r} \tag{6}$$

A control law to cancel undesired nonlinear behaviour is developed as follows:

$$\vec{u} = \mathbf{K_r}\vec{r} - \Theta f(\vec{x}) \tag{7}$$

Using a Lyapunov function, the adaptive law for the control gains K_r and Θ are found to be:

$$\dot{\Theta} = -e^T \mathbf{B_m}\gamma_\theta f(\vec{x})\text{sgn}(b) \tag{8}$$

$$\dot{K}_r = e^T \mathbf{B_m}\gamma_\mathbf{r} r\text{sgn}(b) \tag{9}$$

where e represents the error between the reference model and estimated state vector, and $\mathbf{B_m}\gamma_\theta$ and $\mathbf{B_m}\gamma_\mathbf{r}$ act as adaptive gains. Equations 8 and 9, once implemented into software, are meant to be integrated internally and then fed to the inner control loop.

Three different reference models were determined by modeling the system dynamics for two platforms using Simulink and linearizing the system about different trim points (for translational motion about only x, only y, and rotational motion). In this way, the idea is to implement a regulator using MRAC as an inner loop, where a set point is reached by using one of 3 different modes: 2 for translational motion and one rotational motion. Simulation results for two of the three different modes are found in Figure 7a and 7b.

There is hesitation in accepting these results for face value, as they show the system behaviour matching the reference model in less than a second. Tracking the desired model's states in less than a second is unrealistic given the discrete nature of the physical system, i.e., actuators/sensor data/controller calculations are limited by their operational frequency and that of the on-board processor. Future work involves discretizing the model used in simulation, in addition to creating a higher fidelity model by adding actuator and sensor models.

Given the known limitations of the current MRAC designed, it was decided against implementing MRAC during initial testing of aggregation behaviour, as it would provide an additional unknown variable to the reconfiguration algorithms that are to be discussed later in this paper. For this reason, the simple P-D controller previously discussed was implemented on the Satbots. Upon validation of the reconfiguration algorithms, and possible improvements to the designed MRAC, through the creation and use of a higher fidelity model, MRAC shall be implemented in the Satbots and tested. The rest of the GNC and reconfiguration algorithms, which are now to be discussed, would still apply.

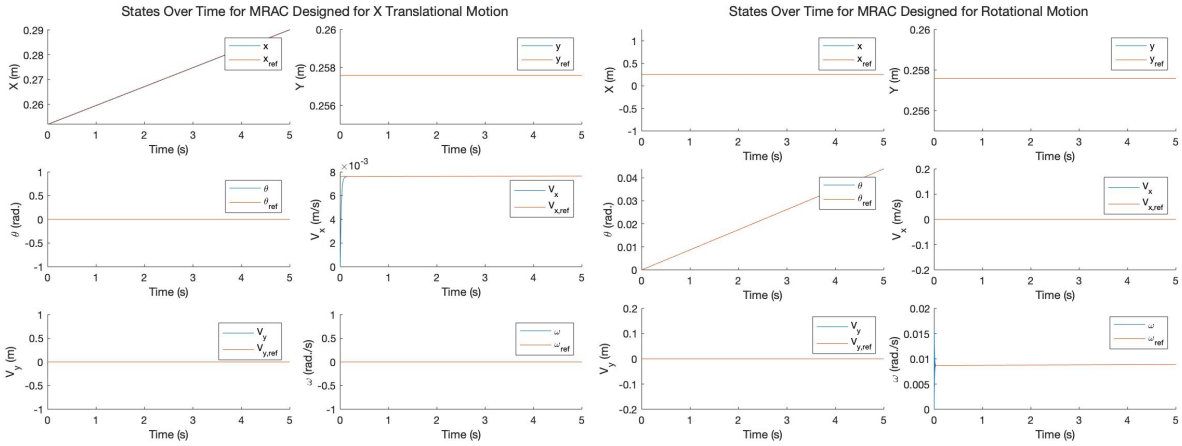
3.5 Thruster Mapping Matrix

After determining the necessary control input for the Satbot to achieve the desired state, the GNC algorithm computes the required firing times for each thruster. However, controlling multiple thrusters on different Satbots simultaneously requires the concept of a scalable thruster mapping matrix, which translates the physical contribution from each thruster into a mathematical model. [6] [11] The diagram in Figure 8 visualizes our approach to orient forces and torques equal about each axis for simplicity.

To simplify several different thrusters and orientations, each force contribution is collected into a simple table, as seen in Table 1.

Table 1: Directional force contributions from each thruster for the Satbot

Thruster	Resultant Body-Axis Force			
	$+\hat{x}$	$-\hat{x}$	$+\hat{y}$	$-\hat{y}$
1		**x**		
2	**x**			
3			**x**	
4				**x**
5	**x**			
6		**x**		
7				**x**
8			**x**	



(a) Results of Simulation ran for 5 seconds for MRAC designed for translational motion in x.

(b) Results of Simulation ran for 5 seconds for MRAC designed for rotational motion.

Fig. 7: Results for two of the three different modes of the MRAC designed.

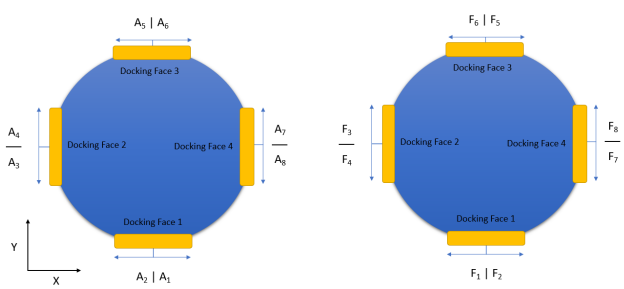


Fig. 8: An illustration of the top view of the Satbot model is presented here. In both diagrams, the yellow rectangles represent the docking face mechanisms. The left image depicts the actuators' thrust output as the arrows extending tangentially from the Satbot's circumference. The right image depicts the corresponding resultant force from each thruster.

Furthermore, firing a single thruster generates a body-axis torque. Since the thrusters are mounted on the edge of the Satbot's circular base, determining each thruster's torque calculation is straight forward. We can map the Satbot's body-axis torque similarly to Table 1. When combining the body-axis force (excluding the z-component) and body-axis torque contributions (about the z-axis) from each thruster, we create its thruster mapping matrix, as seen in Equation 10. Here, the first two rows represent the forces in x and y components respectively, and the third row represents the torque components. Each column corresponds to the particular thruster of that current value; e.g., column one corresponds to thruster one. The benefit to this mapping matrix lies in its scalability, since incorporating more or less thrusters will simply change the number of columns. Additionally, the thruster mapping matrix can be modified accordingly to the location and orientation of each thrusters.

$$\mathbf{M} = \begin{bmatrix} -1 & +1 & 0 & 0 & +1 & -1 & 0 & 0 \\ 0 & 0 & +1 & -1 & 0 & 0 & -1 & +1 \\ -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \end{bmatrix} \quad (10)$$

Another benefit to the thruster mapping matrix arises in its conceptual simplicity in devising autonomous control as the Satbot will understand which thrusters to actuate. Continuing with Figure 8, if the Satbot identifies that it must move *only* in the positive x-direction (relative to its body frame), then the thruster mapping matrix dictates that the Satbot must use both thrusters two and five. Firing individual thrusters contribute to a resultant rotation, which may not be desirable in achieving the final state. Therefore in this example, the system must identify two (or more) thrusters that generate positive force contributions while offsetting each other's torque contribution. With this concept, we form the equation that demonstrates how the control output is achieved by the set of thrusters, as seen in Equation 11.

$$\vec{u} = \mathbf{M}\vec{f} \quad (11)$$

The control output vector, thruster mapping matrix, and firing forces vector are represented by $\vec{u}$, $\mathbf{M}$, and $\vec{f}$ respectively.

The Satbot needs to solve for the necessary forces and torques to be supplied by each thruster to achieve the desired control output, which is previously determined by the selected controller scheme. Since $\mathbf{M}$ may be a non-square matrix, the pseudoinverse, denoted as $\mathbf{M}^{+}$, is computed to find the appropriate firing times. This is shown in Equation 12. A scaling factor of two is applied to account for both the positive and negative contributions of the thrusters. [11] In essence, if a force is requested in the positive x-direction, thruster two and thruster five would be activated, where each thruster provided half the requested force. Additionally, scaling the firing forces vector allows the Satbot to ignore any negative force components from the calculation. Logically in terms of the solenoid valve's unidirectional output, a given thruster cannot contribute a negative force, but rather a positive force in its fixed directional output.

$$\vec{f} = 2\mathbf{M}^{+}\vec{u} \tag{12}$$

3.6 Pulse-Width Modulation

Since the thrusters actuate in an ON/OFF manner, a modulation scheme must be implemented to perform the maneuvers. [5] Though a variety of modulators may achieve the proper control, like Schmitt Triggers and sigma-delta modulators, we opt for the common pulse-width modulator (PWM) to enable Satbot functionality for ease of initial testing. [12]

The PWM converts the requested, non-negative firing forces to solenoid valve opening times. This results in impulses for each thruster for a defined actuation interval or duty cycle. This active time for a given impulse can be determined from Equation 13 and is represented by $T_{on,i}$.

$$T_{on,i} = \frac{f_i}{F_{max}} t_{PWM} \tag{13}$$

The maximum force output represents the physical capability of the thruster and is denoted as F_{max}. The PWM pulse duration is t_{PWM} and is determined by the duty cycle and total modulation period. Each component of the requested firing forces is denoted by f_i.

Additionally, if the determined firing time is less than the minimum opening time of the solenoid valve (approximately 10 ms), then that thruster's contribution is set to zero. If the requested force exceeds the maximum force capability that the thruster can supply, then the firing time is constrained to the maximum pulse duration.

3.7 Aggregation Algorithm Development and the N-Case Configuration

The concept of multi-satellite aggregation introduces the importance of appropriate subsystem reconfiguration, in which the space system must manage N-number of subsystems while adhering to the fundamentals of spacecraft operations. When considering the GNC subsystem, new questions appear regarding the logistics of operation. Firstly, how is spacecraft control ensured for varying configurations and mass properties? Will all actuators remain necessary to function; if not, how can the aggregate system automatically determine which actuators to maintain and disable?

With the framework of the aggregation computational architecture, we devise an algorithm to address the aggregate system behaviour of autonomous GNC reconfiguration. To calculate the aggregate properties for the system, i.e., the new center of mass, moments of inertia, and thruster mapping matrix, we need a system to account for the parameters of N-number of Satbots. Therefore, for each Satbot, we define "spacecraft identification file," or a collection of its GNC components relative to its body axis. Upon aggregation, each Satbot sends its file into the data transport layer of the computational architecture to support the algorithm as needed. For a given Satbot, the spacecraft identification file is comprised of the parameters seen in Table 2. The current table is not comprehensive, and more parameters may be added.

Table 2: The identification file parameters for each Satbot

Number	Parameter
1	Spacecraft Identifier
2	Docking Status Matrix
3	Mass
4	Moments of Inertia
5	Docking Face Position Matrix
6	Sensor Position Matrix
7	Thruster Position Matrix
8	Force Contribution Matrix
9	Torque Contribution Matrix
10	Thruster Mapping Matrix
11	Direction Cosine Matrix

When aggregation occurs, each Satbot exports its identification file through the data transport layer to build a system level "aggregated" identification file, which encompasses all actively docked Satbots. Per the concept of operations described in the early section, the aggregate identification file is distributed to the operating Satbot to continue guidance and

control. However, to model the space system dynamics accurately for GNC, the mass properties must be accounted as any instance of thruster misalignment will result in improper control. Therefore, the challenge in GNC reconfiguration depends on the center of mass and moments of inertia, parameters which continually change upon additional aggregation. Ultimately, the devised algorithm must maintain robustness regardless of the number of components to achieve true autonomy.

First, the algorithm must determine the system's aggregate mass and center of mass. The aggregate mass is simply the sum of all current Satbot masses, as depicted in in Equation 14.

$$M_{agg} = \sum_{i=1}^{N} m_i \tag{14}$$

Then, the center of mass calculation is represented in Equation 15. Denoted in Equation 15 is the center of mass relative to reference frame A; this is not typically the case, as the decentralized architecture allows for any Satbot's body frame to act as the reference point. In this case, when multiple docked Satbots have differently aligned body frames, as visualized in Figure 9, transforming between body frames must be accomplished through the use of the direction cosine matrix.

$${}^{\mathrm{A}}\vec{r_{cg}} = \frac{1}{M_{agg}} \sum_{i=1}^{N} m_i^A \vec{r_{cg,i}} \tag{15}$$

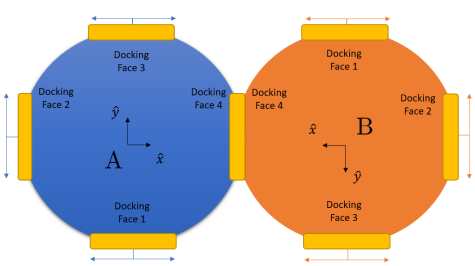


Fig. 9: An illustration of two distinct Satbots, docked with varying orientations. Since parameters are specific to each Satbot's body frame, a transformation to the appropriate frame is required.

As each Satbot measures its current orientation through quaternions, the direction cosine matrix can be computed. We note that the Pozyx board measures the quaternions relative to the set inertial frame; simple matrix mathematics allow us to derive the transformation between two different Satbot frames. Considering the direction cosine matrix, denoted generically as $\mathbf{C}$, the center of mass equation can be expanded as seen in Equation 16. This generically represents the center of mass relative to the A-frame, though this idea is applicable for any selected reference.

$$\begin{aligned} {}^{A}\vec{r_{cg}} = \frac{1}{m_{agg}} [m_A^A \vec{r_{cg,A}} + m_B({}^{\mathrm{A}}\vec{r}_{DF,A} - {}^{\mathrm{A}}\mathbf{C}^{\mathrm{B}\,\mathrm{B}}\vec{r}_{DF,B}) + \\ \ldots + m_N {}^{\mathrm{A}}\vec{r_{cg,N}}] \end{aligned} \tag{16}$$

Similarly, the moments of inertia change with each additional Satbot, which can be determined by using the general form for the parallel axis theorem in Equation 17. [9] Here, $\mathbf{I}_{\mathrm{agg}}$ represents the aggregate moment of inertia tensor. $\mathbf{I}_{\mathrm{cg,\,i}}$ represents the inertia tensor for a given Satbot. $\mathbf{E}_3$ is the identity matrix. From before, the Satbot reference frames may differ, hence the transformation of the inertia tensor must be applied as shown in Equation 18.

$${}^{\mathrm{A}}\mathbf{I}_{\mathrm{agg}} = \sum_{i=1}^{N} {}^{\mathrm{A}}\mathbf{I}_{cg,i} + m_i[({}^{\mathrm{A}}\vec{R}_{cg,i} \cdot {}^{\mathrm{A}}\vec{R}_{cg,i})\mathbf{E}_3 - {}^{\mathrm{A}}\vec{R}_{cg,i} \otimes {}^{\mathrm{A}}\vec{R}_{cg,i}] \tag{17}$$

$${}^{\mathrm{A}}\mathbf{I}_{cg} = ({}^{\mathrm{A}}\mathbf{C}^{\mathrm{B}})({}^{\mathrm{B}}\mathbf{I}_{cg,B})({}^{\mathrm{A}}\mathbf{C}^{\mathrm{B}})^T \tag{18}$$

Combining Equations 17 and 18 expands the parallel axis theorem to become Equation 19 for N-number of Satbots.

$$\begin{aligned} {}^{\mathrm{A}}\mathbf{I}_{\mathrm{Agg}} = {}^{\mathrm{A}}\mathbf{I}_{cg,A} + m_A[({}^{\mathrm{A}}\vec{R}_{cg,A} \cdot {}^{\mathrm{A}}\vec{R}_{cg,A})\mathbf{E}_3 \\ - {}^{\mathrm{A}}\vec{R}_{cg,A} \otimes {}^{\mathrm{A}}\vec{R}_{cg,A}] + \ldots + ({}^{\mathrm{A}}\mathbf{C}^{\mathrm{i}})({}^{\mathrm{i}}\mathbf{I}_{cg,i})({}^{\mathrm{A}}\mathbf{C}^{\mathrm{i}})^T \\ + m_i[({}^{\mathrm{A}}\vec{R}_{cg,i} \cdot {}^{\mathrm{A}}\vec{R}_{cg,i})\mathbf{E}_3 - {}^{\mathrm{A}}\vec{R}_{cg,i} \otimes {}^{\mathrm{A}}\vec{R}_{cg,i}] \end{aligned} \tag{19}$$

Lastly, the change in the center of mass and the addition of thrusters will modify the thruster mapping matrix. Since the Satbots are designed with eight similar thrusters, the size of $\mathbf{M}$ simply becomes [3x8N] for N-number of Satbots. More importantly, the force and torque contributions change according to the selected reference frame and the thruster distances from the new center of mass. Therefore, the appropriate direction cosine matrix must transform the relative thruster position and directional force output vectors. As a result, the new torque contributions from each thruster must be recalculated.

Another important consideration requires that the thrusters adjacent to the active docking faces deactivate to prevent potential damage or interference to any Satbot. Pairing the aggregated docking status with **M** allows the Satbots (and thus future space systems) to identify which thrusters to disable. With the reconfiguration algorithm and GNC subsystem developed through software, we connect its application with hardware to test the first instance of demonstrating the aggregation behaviour.

3.8 Test Case

In testing the reconfiguration algorithm, we sought to demonstrate the following:

1. Sufficient GNC response and maneuvering to the desired state
2. Proper real time data transfer between multiple Satbots
3. Proper thruster history for an aggregate system after identifying active and deactivated thrusters

The first metric assesses the Satbot's ability to perform the first stage of autonomous GNC, which is imperative to operating any number of Satbots. The second metric addresses the capability to operate an aggregated system. The third metric signals the success in reconfiguring the GNC by exhibiting the proper thruster actuation.

For initial demonstration, two Satbots are fixed together with velcro at the docking port to emulate an aggregation for reconfiguration testing. Each Satbot is pre-configured with the necessary aggregate identification file parameters that feed into the algorithm.

As the current algorithm does not account for continuous varying mass, both Satbots are set to the same wet mass to ensure that the aggregate center of mass calculation properly defines this location. Considering the concept of the thruster mapping matrix, adding more thrusters equally divides the required force and torque contributions. In this case, propelling the aggregate system forward would activate the appropriate thrusters from both Satbots. Given that the Satbots begin at the same wet mass, both Satbots will maintain equal masses throughout operation, and the aggregate center of mass does not change.

Once docked, one of the two Satbots will be selected to perform the bulk of the GNC operation and reconfiguration algorithm. The other Satbot continuously loops through a receiver function that accepts the calculated aggregate firing times and actuates its thrusters. The GNC operation continues until the aggregate center of mass approaches the final desired state. Though the full-scale computational architecture has not been fully developed, we utilize several communication protocol to support the first stages of aggregation testing.

3.9 Communication Protocol in Firing Time Transfer

The two standard communication protocols used to transfer data between devices are TCP and UDP. Each of these protocols have their advantages and disadvantages, and the choice between them is dependent on the constraints of data being transferred and its importance to the overall system. TCP is a reliable transport layer protocol, as such TCP guarantees that the data packet that is sent and received by the communicating devices are in order and error free. The draw-back of TCP is in the overhead that it introduces in order to ensure reliability, not only does it increase the data packet size (for all the error checking and packet ordering) but it also reduces the effective through put. For small aggregations this may be admissible as the number of hardware devices are few, as such the amount of data to be passed will be small. But as the aggregation scales, the reduced bit rate and reliability guarantee could potentially reduce the overall responsiveness of the overall aggregation. UDP on the other hand is an unreliable transport layer protocol, it makes no guarantee on the order of the data packets nor any errors in the data packets. The lack of any guarantees reduces the overhead in each data packet that is sent, allowing for a higher throughput of data. Given these properties, UDP provides the speeds and throughput needed to scale the aggregation, but lack of guarantee poses a huge drawback. In order to preserve both speed and reliability, another protocol is needed. Asbsuch RUDP would provide the best balance between TCP and UDP for future development. For the purpose of proof of concept and lower development time, UDP has been used for the transport layer.

As we develop and test the physical aggregation of the Satbots, we seek additional validation of the aggregation model's GNC operation through simulation.

3.10 Simulation

Currently, two different simulations have been utilized for the cases of $N = 1$ and $N = 2$ Satbots. The first involves a simulation built using Simulink, in which the plant was modeled using mathematical equations of the expected system dynamics, and was primarily used to design the MRAC

adaptive controller. The second is a NX Motion/Simulink co-simulation. This simulation involves using NX Motion for the system's dynamics (using the Satbot's CAD properties) and Simulink for the controller design. Once the NX Motion simulation is designed with the desired control inputs and outputs, a solution can be created in NX Motion, which produces a corresponding Matlab script. By running this script, a NX Motion Plant block (a Simulink mask) is created in Simulink. A feedback system can be created in Simulink and when the simulation is ran, NX Motion uses control inputs determined in Simulink and feeds its sensor outputs back to Simulink for every timestep. Once the simulation is complete, the results can be animated using NX Motion, providing a visualization tool.

The second of these two simulations will prove especially useful in validating the GNC algorithms previously described throughout this paper for cases where $N > 3$, as testing with the Satbots is limited by hardware and resources. With the current work presented in this paper, the co-simulation was used to test some of the GNC algorithms, particularly the thruster mapping matrix. Results from this testing showed the need to modify the mapping matrix for a more general solution. In this way, when the mapping matrix was used in Simulink to determine the appropriate control inputs, it resulted in unexpected rotation. The algorithm can be modified with a simple fix in which the distances to each thruster from the CG is accounted for, as seen in Equation 20.

$$\mathbf{M} = \begin{bmatrix} -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ -d_1 & d_2 & -d_3 & d_4 & -d_5 & d_6 & -d_7 & d_8 \end{bmatrix} \tag{20}$$

When the pseudoinverse of this matrix was determined using the thruster distances measured in the CAD model and implemented into the co-simulation, it resulted in appropriate system behavior, given requests corresponding to non-rotational motion. Implementation of the modified mapping matrix is to be considered for possible future work.

4 Future Work

The future of the Satbot focuses on developing the overarching architecture, the control system's adaptability, and maintaining valid spacecraft design while facilitating N-number aggregations. Current considerations include the:

- Construct the framework of the computational architecture.
- Potential to refine simulation and design Self-Tuning Control (STC) and compare performance.
- Implement and test MRAC on platforms.
- Further developments to the simulation tools for N-number of Satbots.
- Fabricating the next generation of Satbots (Figure 10).

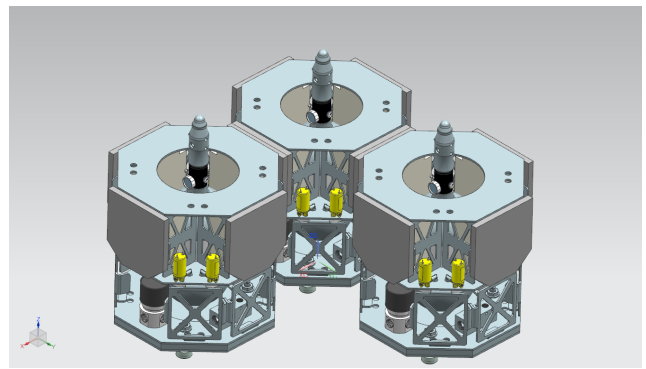


Fig. 10: The proposed design of the Satbot. This new model aims to facilitate four docking ports, decrease the system mass, and improve daily operation.

5 Conclusion

The value proposition of cellularization to enable aggregation on orbit depends upon some sort of computational architecture that enables ubiquitous connections. USC SERC identified two elements to a new architecture, the transport and element identifications, and behaviours for an aggregated system. This paper took the first step to demonstrate one space system function that is required to be maintained during aggregation, GNC. The preliminary concepts of an autonomous GNC subsystem and reconfiguration algorithm and simulation results were detailed through the development of a simple Satbot that represented a 3DOF thruster-based control satellite in a test facility. The current testing shows direct results that proves the possibility to aggregate two disparately controlled Satbots and operate as a single monolithic entity. The architecture identified showed a possible vector that can apply to N cells/elements/components/satlets on orbit, and further research and simulation to show tens to hundreds of elements to explore the boundaries of the various layers will be done.

6 Acknowledgements

We wish to thank the Information Sciences Institute for sponsoring the research detailed in this paper. We would also like to show our appreciation to the researchers and students at the Space Engineering Research Center for their dedication, collaboration, and efforts in assisting the development of this project. In particular, we would like to thank Monica Campos for the development of thruster quantification tests and the results used for our pulse width modulation and Michael Smat who is currently working on the redesign of the Satbots.

References

[1] Aswin R. Arya, V. and Ashni Elisa George. Modified model reference adaptive control for the stabilization of cart inverted pendulum system. *International Research Journal of Engineering and Technology (IRJET)*, 5(4).

[2] Hill L. Fowler E. Hoag L. Sullivan B. Will P. Barnhart, D. A Market for Satellite Cellularization? A first look at satlet morphology's implementation and potential impact on the space industry. In *AIAA Space 2013 Conference & Exposition*, page 5486.

[3] Hill L. Turnbull M. Barnhart, D. and P. Will. Changing Satellite Morphology through Cellularization. In *AIAA Space 2012 Conference & Exposition*, page 5262.

[4] L. Hill. Fowler E. Hunter R. Hoag L. Sullivan B. Barnhart, D. and P. Will. A Further Look at Potential Impact of Satlets on Design, Production, and Cost of Satellite Systems. In *Small Satellite Conference 2014*, pages 14–V–6.

[5] Allen Chen. Propulsion system characterization for the spheres formation flight and docking testbed. Master's thesis, Massachusetts Institute of Technology, 2002.

[6] Mark Ole Hilstad. A multi-vehicle testbed and interface framework for the development and verification of separated spacecraft control algorithms. Master's thesis, Massachusetts Institute of Technology, 2002.

[7] iBoss GMBH. iSSI. `http://www.iboss-satellites.com/fileadmin/Templates/iBOSS_Satellites/Media/iSSI.pdf`, 2019. [Online; accessed 30-September-2019].

[8] Mirczak W. Jaeger, T. and B. Crandall. Cellularized Satellites - A Small Satellite Instantiation that Provides Mission and Space Access Adaptability. In *Small Satellite Conference 2016*.

[9] Christopher M Jewison, Bryan McCarthy, David C Sternberg, Daniel Strawser, and Cheng Fang. Resource aggregated reconfigurable control and risk-allocative path planning for on-orbit servicing and assembly of satellites. In *AIAA Guidance, Navigation, and Control Conference*, page 1289, 2014.

[10] Nhan T. Nguyen. *Model Reference Adaptive Control*. Springer International Publishing.

[11] Richard Zappulla, Josep Virgili Llop, Hyeongjun Park, Costantinos Zagaris, and Marcello Romano. Floating spacecraft simulator test bed for the experimental testing of autonomous guidance, navigation, control of spacecraft proximity maneuvers and operations. *AIAA/AAS Astrodynamics Specialist Conference*, Sep 2016.

[12] Richard Zappulla, Josep Virgili-Llop, and Marcello Romano. Spacecraft thruster control via sigma–delta modulation. *Journal of Guidance, Control, and Dynamics*, 40(11):2928–2933, 2017.